

Konstantinos Poularakis, George Iosifidis, Georgios Smaragdakis,  
Leandros Tassiulas

# Optimizing Gradual SDN Upgrades in ISP Networks

**Journal article** | **Accepted manuscript (Postprint)**

This version is available at <https://doi.org/10.14279/depositonce-9375>



Poularakis, K., Iosifidis, G., Smaragdakis, G., & Tassiulas, L. (2019). Optimizing Gradual SDN Upgrades in ISP Networks. *IEEE/ACM Transactions on Networking*, 27(1), 288–301.  
<https://doi.org/10.1109/tnet.2018.2890248>

## Terms of Use

© 2019 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

**WISSEN IM ZENTRUM**  
**UNIVERSITÄTSBIBLIOTHEK**

Technische  
Universität  
Berlin

# Optimizing Gradual SDN Upgrades in ISP Networks

Konstantinos Poularakis<sup>1</sup>, George Iosifidis<sup>2</sup>, Georgios Smaragdakis<sup>3</sup>, and Leandros Tassioulas<sup>1</sup>

<sup>1</sup>Department of Electrical Engineering and Yale Institute for Network Science, Yale University, USA

<sup>2</sup>School of Computer Science and Statistics, and CONNECT, Trinity College Dublin, Ireland

<sup>3</sup>Chair of Internet Measurement and Analysis, TU Berlin, Germany

**Abstract**—Nowadays, there is a fast-paced shift from legacy telecommunication systems to novel Software Defined Network (SDN) architectures that can support on-the-fly network reconfiguration, therefore, empowering advanced traffic engineering mechanisms. Despite this momentum, migration to SDN cannot be realized at once especially in high-end networks of Internet Service Providers (ISPs). It is expected that ISPs will gradually upgrade their networks to SDN over a period that spans several years. In this paper, we study the SDN upgrading problem in an ISP network: which nodes to upgrade and when. We consider a general model that captures different migration costs and network topologies, and two plausible ISP objectives: (1) the maximization of the traffic that traverses at least one SDN node, and (2) the maximization of the number of dynamically selectable routing paths enabled by SDN nodes. We leverage the theory of submodular and supermodular functions to devise algorithms with provable approximation ratios for each objective. Using real-world network topologies and traffic matrices, we evaluate the performance of our algorithms and show up to 54% gains over state-of-the-art methods. Moreover, we describe the interplay between the two objectives; maximizing one may cause a factor of 2 loss to the other. We also study the *dual* upgrading problem, i.e., minimizing the upgrading cost for the ISP while ensuring specific performance goals. Our analysis shows that our proposed algorithm can achieve up to 2.5 times lower cost to ensure performance goals over state-of-the-art methods.

**Index Terms**—Software Defined Networks, Gradual Deployment, ISP Networks.

## I. INTRODUCTION

**Motivation.** Software Defined Networking (SDN) [2] enables unprecedented network management flexibility through the separation of the network control and data planes, and the centralization of the former in designated network entities referred to as controllers. A controller maintains a global view of the network state, including network topology, traffic load, and link failures, and can leverage this information to dynamically select the routing paths for each network flow. This approach departs significantly from traditional IP protocols, like Open Shortest Path First (OSPF) [3], that are destination-based and route traffic along shortest paths using static link weight metrics. SDN, therefore, empowers

advanced Traffic Engineering (TE) mechanisms that can respond on-the-fly to network changes and support fine-grained routing decisions per flow. Today, many Internet Service Providers (ISPs) rely on Multiprotocol Label Switching (MPLS) [4] to achieve such flexibility and steer traffic without being constrained by shortest paths. Nevertheless, MPLS has a number of shortcomings, e.g., it relies on pre-determined source-destination paths that are hard to maintain and time-consuming to re-configure. On the other hand, with SDN, a central controller can be used to change network configuration in almost real-time and achieve per-flow QoS objectives such as end-to-end delay and end-to-end bandwidth. This makes SDN a particularly attractive technology.

However, as it happens with most novel network protocols and architectures [5], migration to SDN cannot be realized at once. This is particularly true for the large and expensive core networks of ISPs. Namely, the one-step SDN upgrade of entire ISP networks is practically impossible since it poses an enormous operational burden, and also raises performance and security risks [6]. On top of that, such upgrades require huge capital expenditures since network components (e.g., backbone routers) are very expensive. Besides, upgrading newly installed legacy routers is economically prohibitive<sup>1</sup>. Given the above, it is expected that ISPs will opt to migrate to SDN incrementally, i.e., by gradually upgrading their network nodes over a period that spans several years. In these incremental SDN deployments, the controllers will manage only the SDN-enabled nodes, while the remaining legacy network will still use OSPF-like routing protocols.

Even in such *hybrid SDN* networks, the ISPs can accrue important benefits. Namely, for the traffic that crosses at least one SDN node, it is possible to apply various sophisticated policies such as access control, firewall actions, and other middlebox-supported in-network services [8]. Moreover, using the SDN nodes it is possible to dynamically control the routing path of the flows by overriding the underlying legacy OSPF or MPLS protocol and thus, create a more flexible network [9]. In line with prior works, we will use the term *programmable traffic* to differentiate the traffic that

Part of this work was appeared in the proceedings of IEEE Infocom 2017 [1]. K. Poularakis acknowledges the Bodossaki Foundation, Greece, for a postdoctoral fellowship. G. Iosifidis was supported by a research grant from Science Foundation Ireland (SFI) under Grant Number 17/CDA/4760. G. Smaragdakis was supported by the ERC Starting Grant ResolutioNet (ERC-StG-679158). L. Tassioulas was supported by the National Science Foundation under Grant CNS 1815676.

<sup>1</sup>A typical router replacement window is 3 to 5 years; more importantly a network's routers have out-of-phase cycles, i.e., need to be replaced in different times, e.g., see *Lifecycle Financing from Cisco Capital*. Also, router costs vary significantly from a few tens of thousands of dollars to more than \$100K [7].

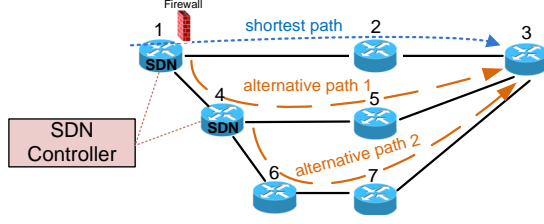


Fig. 1. A network that is partially upgraded to SDN. The two SDN nodes can act as firewalls or dynamically control the routing path.

traverses at least one SDN node from that not traversing any SDN nodes. Both in-network services and the availability of alternative routing paths (that can be dynamically selected) are extremely useful for ISPs. Besides, if the flow crosses more than one SDN nodes, the ISP has even more dynamic routing options and hence can further increase the *TE flexibility* of its network.

Let us show the potential of this approach with a simple example. Consider the hybrid SDN network shown in Figure 1 that routes a flow from source node 1 to destination node 3. Here, only two of the seven nodes are upgraded with SDN capabilities (nodes 1 and 4). Using OSPF, the flow is always routed along the shortest path. However, node 1 can dynamically decide to drop (instead of forwarding) the packets, acting, e.g., as a firewall. It can also override the OSPF shortest path by routing the packets through node 4. The packets will then follow the alternative path 1 which is the OSPF shortest path connecting node 4 with 3. Such flow rerouting is important when a link of the shortest path fails or becomes temporarily congested. Since node 4 is also upgraded to SDN, it can similarly defer packets towards alternative path 2. In other words, *as the number of SDN-enabled nodes increases, the set of alternative paths increases as well. Hence, there exist more degrees of freedom (or, flexibility) in performing dynamic TE.*

To gain the maximum benefits, it is important to identify which SDN upgrade schedule is suitable for a given network. Namely, every ISP needs to carefully select which nodes to upgrade, and when exactly to do so. Especially this latter aspect of timing has many implications. First, like every new technology, the initial high cost of SDN decreases with a high pace over time [10]. Hence the ISPs face a dilemma of early upgrade that will allow them to reap the new technology benefits immediately and a slow upgrade that will reduce their capital expenditures. More practical, the ISPs need to decide how many nodes to upgrade in each period, which for ISPs usually amounts to 6-12 month intervals. Second, the routers are highly heterogeneous since they serve a different amount of traffic and have a different remaining lifetime, and this further perplexes these decisions.

In summary, every ISP has to address the following two questions: (i) *How many nodes to upgrade in each period? Should it upgrade all nodes as early as possible or wait for the prices to fall?*, and (ii) *After deciding the number of nodes to be upgraded, which specific nodes to select?* The ISP's goal might be to maximize the volume of

programmable traffic or the TE flexibility by increasing the dynamically selectable alternative paths, based on the ISP's priorities and preferences. Despite the very important recent prior works on hybrid SDN networks, e.g., see [9], [11] and Section VIII for a detailed overview, we currently lack a systematic understanding regarding the above issues. Therefore, our goal in this work is to investigate policies for SDN upgrade scheduling in large (and expensive) operational ISP networks, and focus mainly on the impact of time-dimension and the interplay between traffic programmability and TE flexibility benefits.

**Methodology and Contributions.** We develop a methodology to address the above two questions posed by ISPs regarding SDN migration. We introduce a model of SDN upgrades general enough to capture different migration costs, as well as ISP topologies and traffic demands. We then utilize this model to derive the optimal scheduling for router upgrades in the ISP network over a period that may span several years. We consider two ISP objectives. First, we target the maximization of the programmable traffic, i.e., the traffic that traverses at least one SDN node (*Obj1*). This upgrading policy, if designed properly, can have significant benefits [9], [11], since it allows an ISP to control how the traffic flows in its own network. The second objective (*Obj2*) aims to maximize the TE flexibility. This objective is achieved by increasing the number of alternative paths through the SDN upgrades. For each one of the two objectives, we formulate a rigorous optimization problem and devise the desirable SDN upgrading policy (or, schedule): which nodes to upgrade and when.

In both cases, finding the upgrading policy requires the solution of challenging combinatorial optimization problems. Namely, we show that for *Obj1* this problem is NP-Hard even to approximate to any factor better than  $1 - 1/e$ . For the special case in which all the node upgrades take place at the same time period, we show that a modified version of a classic greedy algorithm, which enumerates all possible triplets of nodes as candidate solutions, achieves the best possible approximation ratio. We also show a simple way to extend this algorithm for the general case where the node upgrades can take place at different time periods. We also present a second class of more sophisticated algorithms with improved approximation ratios by expressing *Obj1* as the maximization of a submodular set function [12], i.e., a function that satisfies the diminishing returns property.

Then, we study *Obj2* (maximizing TE flexibility). This is a more complex problem which can be expressed as the maximization of a function with bounded supermodular degree [13]. Using this result, we present another greedy-based algorithm that approximately solves this problem. For the sake of completeness, we also consider the “dual” version of the upgrading problem (*Obj3*), where the above objectives are treated as constraints and subject to them we minimize migration costs. For a simple, yet practical, case an approximation algorithm is proposed using a binary search technique.

We evaluate the performance of the proposed algorithms using two datasets of real network topologies and traffic matrices [14], [15]. The results clearly differentiate situations in which upgrades should be spread over many instead of one-time step.

The contributions of this work are summarized as follows:

- *SDN Upgrading Problem.* We introduce the problem of gradually (and partially) upgrading an ISP network to SDN, using general models of costs and different objectives. The upgrades can take place at different time periods, introducing different costs at each period due to technology maturity, the different life-cycle of the network equipment and other practical limitations.
- *Maximizing Programmable Traffic (Obj1).* For the programmable traffic maximization objective, we show that the SDN upgrading problem is NP-Hard to approximate to any factor better than  $1 - 1/e$ . Then, we present a simple algorithm matching this factor for the special case of one time period and show how it can be extended for the general case. We also present additional more sophisticated approximation algorithms using the theory of submodular functions.
- *Maximizing TE Flexibility (Obj2).* For the objective of maximizing TE flexibility through the availability of SDN-enabled routing paths, we show that the optimization problem is more complex. We present an approximation algorithm by expressing it as the maximization of a function with bounded supermodular degree.
- *Minimizing migration costs (Obj3).* For the “dual” problem of minimizing migration costs, we show that it is fundamentally different from the above problems. We also present an approximation algorithm using a binary-search technique.
- *Dataset-driven Evaluation.* We evaluate the proposed algorithms using real-world network topologies and traffic matrices. We find that our approach can increase by 54% the amount of programmable traffic compared to two state-of-the-art methods in practical scenarios. We also find that by optimizing Obj1, benefits are also realized for Obj2 (and vice versa) and we explore the interplay between the two objectives. Finally, we show that our proposed algorithm for Obj3 can achieve up to 2.5 times lower cost to ensure performance goals over two additional state-of-the-art methods.

The rest of the paper is organized as follows. Section II describes the system model and formalizes the SDN upgrading problem for Obj1. In Sections III and IV, we present theoretical results about the computational complexity of this problem and approximation algorithms. Section V considers Obj2 and presents an approximation algorithm. We tackle the cost minimization version of the problem (Obj3) in Section VI. Section VII presents the dataset-driven evaluation of our proposed algorithms, while Section VIII reviews our contribution compared to related works. We conclude our work in Section IX.

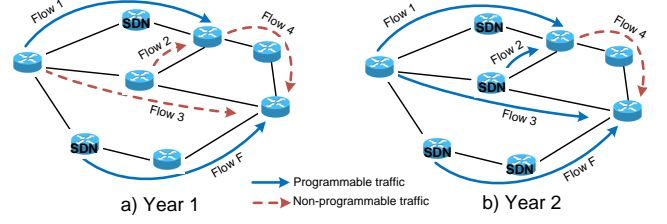


Fig. 2. An example of incremental SDN upgrades for  $T = 2$  years. Two nodes are upgraded each year, increasing the amount of programmable traffic.

## II. MODEL AND PROBLEM FORMULATION

We adopt a general model representing a large ISP core network with a set  $\mathcal{N}$  of  $N$  nodes (e.g., IP/MPLS routers). The network traffic consists of a set  $\mathcal{F}$  of  $F$  origin-destination flows. With traditional IP routing protocols, like OSPF, each flow  $f$  follows the shortest path to the destination. With more advanced protocols, like MPLS, flow  $f$  can follow a different (non-shortest) path based on some source-destination bandwidth declaration mechanism. In any case, we denote by  $\mathcal{N}_f \subseteq \mathcal{N}$  the set containing the nodes along this initial path for flow  $f$ . The ISP may decide to upgrade some of the nodes to SDN, and makes these decisions along a time interval of  $t = 1, \dots, T$  time periods,  $t \in \mathcal{T}$ . Typically, such decisions are made in an annual or semi-annual fashion, and by accommodating the lifetime of this type of equipment (3-5 years). Thus, a usual value can be  $T = 5$  or 10 time periods. An example of incremental SDN upgrades is depicted in Figure 2.

Moreover, the global Internet traffic increases with time, having an expected annual growth rate of 22 percent from 2015 to 2020 [16]. To capture these dynamics, we denote with  $\lambda_{tf}$  (bps) the average rate of flow  $f$  at period  $t$ , where  $\lambda_{tf} \geq \lambda_{t'f}, \forall t > t'$ . Although traffic variations may appear also within the same period, it is expected that in backbone networks with high aggregation of flows the traffic will not to be very volatile. The network topology might dynamically change as well. However, it is expected that the topology of the backbone of a network is less likely to change within a few years, e.g., up to five years. Upgrading a node to SDN requires capital expenditures, e.g., for buying a new SDN-enabled device, installation costs, etc. These costs typically differ across nodes. For example, upgrades to edge nodes are typically less expensive than core network nodes, while it is definitely more cost-efficient to upgrade a node at the end of its lifetime (rather than a newly installed one). Besides, the costs are likely to drop over time as the SDN technology matures. We denote with  $b_{tn}$  (in USD) the cost for upgrading node  $n$  at period  $t$ , where it may be  $b_{tn} \neq b_{t'n'}, n \neq n'$ , and  $b_{tn} \leq b_{t'n'}, t \geq t'$ .

We introduce the optimization variable  $x_{tn} \in \{0, 1\}$  that indicates whether node  $n \in \mathcal{N}$  will be upgraded to SDN at time period  $t$  or not. These variables constitute the *upgrading policy* of the ISP:

$$\mathbf{x} = (x_{tn} \in \{0, 1\} : t \in \mathcal{T} \ n \in \mathcal{N}). \quad (1)$$



We consider the case that the ISP has an available monetary budget  $B$  (in USD) that can invest in SDN upgrades. The ISP may opt to either invest this capital at once or spread the budget over different time periods. In any case, the SDN upgrading policy must satisfy the following constraint:

$$\sum_{t \in \mathcal{T}} \sum_{n \in \mathcal{N}} x_{tn} b_{tn} \leq B. \quad (2)$$

We note that in the above constraint we assume a *fixed* cost  $b_{tn}$  for upgrading node  $n$  at time period  $t$ . In practice, the ISP may obtain a *discounted* price per unit if it orders many SDN nodes at the same time period. Our model can be generalized to capture such cases, e.g., by replacing the above linear constraint with a non-linear constraint. Clearly, each node can be upgraded to SDN at most once:

$$\sum_{t \in \mathcal{T}} x_{tn} \leq 1, \forall n \in \mathcal{N}. \quad (3)$$

Since the network is upgraded incrementally, some flows may still traverse only legacy nodes, or they may traverse a mixture of upgraded and legacy nodes. A flow that traverses at least one SDN node, can be used to realize a programmatic interface, e.g., for implementing a firewall or routing traffic to alternative paths (as we depicted in Figure 1). The volume of *programmable traffic* can be expressed as follows:

$$J_1(\mathbf{x}) = \sum_{t \in \mathcal{T}} \sum_{f \in \mathcal{F}} \lambda_{tf} 1_{\{\sum_{n \in \mathcal{N}_f} \sum_{t' \leq t} x_{t'n} \geq 1\}}. \quad (4)$$

Here,  $1_{\{\cdot\}}$  is the indicator function; it is equal to one if the condition in the subscript is true, otherwise zero.

The first natural objective of the ISP is to find the upgrading policy that maximizes the volume of programmable traffic (*PTM problem*):

$$\begin{aligned} \text{Obj1:} \quad & \max_{\mathbf{x}} \quad J_1(\mathbf{x}) \\ \text{s.t.} \quad & \text{constraints: (2), (3)} \\ & x_{tn} \in \{0, 1\}, \forall t \in \mathcal{T}, n \in \mathcal{N}. \end{aligned} \quad (5)$$

This is a challenging combinatorial optimization problem. For example, in a network of  $N = 100$  nodes and  $T = 10$  time periods there will be  $2^{NT} = 2^{1000}$  possible solutions, and therefore brute force algorithms that exhaustively search the solution space will require prohibitively large running time.

The second plausible ISP objective that we examine (**Obj2**) is the maximization of the TE flexibility. The latter is directly proportional to the number of alternative routing paths that are dynamically selectable through the SDN upgraded nodes. We present the formulation of this objective in Section V.

### III. COMPLEXITY ANALYSIS OF THE PTM PROBLEM

In this section, we investigate thoroughly the *complexity of the PTM problem*, and we propose a simple algorithm with a provable approximation ratio. We begin with the special case of  $T = 1$  period and then extend our results for any  $T$ .

*Theorem 1:* For the special case of  $T = 1$  time period, the PTM problem is NP-Hard, but there exists a polynomial-time  $(1 - 1/e)$ -approximation algorithm.

In order to prove Theorem 1, we will consider a variant of the coverage problem known as *Budgeted Maximum Coverage Problem (BMCP)* [17]. The latter can be defined as follows.

**BMCP:** Given a set of elements  $\mathcal{E} = \{E_1, E_2, \dots, E_l\}$  with associated weights  $\{w_i\}_{i=1}^l$  and a collection of sets  $\mathcal{S} = \{S_1, S_2, \dots, S_m\}$  defined over  $\mathcal{E}$  with associated costs  $\{c_i\}_{i=1}^m$ , the goal is to find a collection of sets  $\mathcal{S}' \subseteq \mathcal{S}$ , such that the total cost of sets in  $\mathcal{S}'$  does not exceed budget  $C$  and the total weight of elements covered by  $\mathcal{S}'$  is maximized.

Then, we will prove the following Lemma.

*Lemma 1:* PTM for  $T = 1$  and BMCP are equivalent problems.

*Proof:* Given an *arbitrary* instance of the BMCP, we construct a *specific* instance of PTM problem for a single period  $t$  as follows. We create a separate node for each set, i.e.,  $\mathcal{N} = \{1, 2, \dots, |\mathcal{S}|\}$ . The upgrading costs of the nodes match the costs of the sets, i.e.,  $b_{tn} = c_{S(n)}$  where the set  $S(n)$  corresponds to node  $n$ . The upgrading budget  $B$  matches the budget  $C$  in the BMCP instance. We also create a separate flow for each element, i.e.,  $\mathcal{F} = \{1, 2, \dots, |\mathcal{E}|\}$ . The flow rates match the weights of the elements, i.e.,  $\lambda_{tf} = w_{E(f)}$  where the element  $E(f)$  corresponds to flow  $f$ .

If there exists a set of nodes which upgrading them to SDN makes  $Q$  amount of traffic to traverse at least one SDN node (programmable traffic), then there will also be a collection of sets that cover elements with total weight  $Q$ . The covered elements will correspond to the flows that traverse the SDN nodes. The total cost of the node upgrades will be equal to the total cost of the sets. Hence, the solution to the PTM problem provides a solution to the BMCP with the same value. Conversely, given a solution to the BMCP problem, one can solve the PTM problem by upgrading the nodes corresponding to the sets picked by the BMCP solution. The two solutions will have the same value. ■

It is known that it is NP-Hard to approximate the BMCP problem to any factor better than  $1 - 1/e$  [17]. Therefore, Lemma 1 indicates that the same statement holds for the PTM problem. Hence, we have proved the first part of Theorem 1. To prove the second part, we leverage an algorithm in the literature that is used for solving the BMCP. Specifically, the work in [17] has shown that a *modification of the well-known greedy algorithm* yields a  $(1 - 1/e)$ -approximate solution to BMCP (matching the best possible factor). When applied to the PTM problem for  $T = 1$ , this algorithm enumerates all subsets of nodes with cardinality 3, that have total cost at most  $B$ . It completes each subset to a candidate solution in a greedy fashion. Specifically, it iteratively upgrades the node with the highest ratio of the traffic that becomes programmable over upgrading cost. Another set of candidate solutions consists of all subsets of nodes with cardinality less than 3, which have total cost at most  $B$ . The algorithm will output the candidate solution

---

**Algorithm 1:** Modified-greedy algorithm for one period  $t$ 


---

```

1  $CS_1 \leftarrow \emptyset$ ;
2 for all subsets  $\mathcal{N}' \subseteq \mathcal{N}$  such that  $|\mathcal{N}'| = 3$  and
    $\sum_{n \in \mathcal{N}'} b_{tn} \leq B$  do
3    $\mathcal{U} \leftarrow \mathcal{N} \setminus \mathcal{N}'$ ;
4   repeat
     select  $n \in \mathcal{U}$  that maximizes
        $\Delta J_1(\mathcal{N}', n)/b_{tn}$ ;
5   if  $\sum_{n \in \mathcal{N}' \cup \{n\}} b_{tn} \leq B$  then
     |  $\mathcal{N}' \leftarrow \mathcal{N}' \cup \{n\}$ ;
     end
6    $\mathcal{U} \leftarrow \mathcal{U} \setminus \{n\}$ ;
   until  $\mathcal{U} = \emptyset$ ;
7   if  $J_1(\mathbf{x}(\mathcal{N}')) > J_1(\mathbf{x}(CS_1))$  then
     |  $CS_1 \leftarrow \mathcal{N}'$ ;
   end
8  $CS_2 \leftarrow \operatorname{argmax} \{ J_1(\mathcal{N}'), \text{ such that } \mathcal{N}' \subseteq \mathcal{N},$ 
    $|\mathcal{N}'| < 3 \text{ and } \sum_{n \in \mathcal{N}'} b_{tn} \leq B \}$ ;
9 if  $J_1(\mathbf{x}(CS_1)) > J_1(\mathbf{x}(CS_2))$  then
   | Upgrade nodes in  $CS_1$ ;
   end
10 Else Upgrade nodes in  $CS_2$ ;

```

---

that achieves the highest programmable traffic volume. The pseudocode of the algorithm is presented in Algorithm 1, and described in the sequel.

Here,  $CS_1, CS_2 \subseteq \mathcal{N}$  are the two candidate solutions found by the algorithm.  $CS_1$  is initialized to the empty set (line 1).  $\mathcal{U}$  denotes the set of nodes that are *not* included in the current candidate solution.  $\Delta J_1(\mathcal{N}', n)$  indicates the additional traffic that becomes programmable when node  $n$  is upgraded, given that the nodes in subset  $\mathcal{N}'$  are already upgraded. The term  $\mathbf{x}(CS_1)$  indicates the vector  $\mathbf{x}$  for which  $x_{nt} = 1$  iff  $n \in CS_1$ . Similarly, for  $\mathbf{x}(CS_2)$ . For each possible triplet of upgraded nodes, the algorithm iteratively upgrades the node with the highest ratio of traffic that becomes programmable over upgrading cost (line 4). Nodes will be skipped if their upgrade violates the budget constraint (line 5). At the end of the loop (line 8),  $CS_1$  will be the solution with the best value.  $CS_2$  will be the best solution of cardinality less than 3. The algorithm will compare  $CS_1$  and  $CS_2$  and pick the solution with the best value (lines 9-10).

With the Modified-greedy algorithm, we have proved the second part of Theorem 1. Nevertheless, this algorithm works only for a single time period ( $T = 1$ ). The rest of this section extends the results for the general case with many time periods.

**Theorem 2:** For the general case, there exists an  $O((1 - 1/e)/\log(T))$ -approximation algorithm to PTM problem.

We will prove this theorem by extending any approximation algorithm that works for one period to many periods. Formally, we prove the following lemma.

**Lemma 2:** We can extend any  $\alpha$ -approximation algorithm for the  $T = 1$  case of the PTM problem, to obtain an  $O(\alpha/\log(T))$ -approximation for the general  $T > 1$  case.

*Proof:* For each time period  $t = 1, 2, \dots, T$ , we define a new problem that has its own budget  $B$  and is independent of time periods other than  $t$ . For the problem at period  $t$ , the upgrading decisions are denoted by  $\mathbf{x}^t = (x_{tn} : n \in \mathcal{N})$ . The objective of this problem is to maximize the programmable traffic at time period  $t$  given by the function  $v(\mathbf{x}^t) = \sum_{f \in \mathcal{F}} \lambda_{tf} \mathbf{1}_{\{\sum_{n \in \mathcal{N}} x_{tn} \geq 1\}}$ . We note that the function  $v(\mathbf{x}^t)$  differs from  $J_1(\mathbf{x})$  in that (i) it considers only the programmable traffic at time period  $t$  and (ii) it neglects the impact of upgrading decisions taken in time periods other than  $t$ .

We denote by  $ALG_t$  and  $OPT_t$  an  $\alpha$ -approximate and an optimal solution to the problem defined for time period  $t$ . Our main idea is to use each  $ALG_t$  solution to construct a solution  $H_t$  for the original (for multiple time periods) PTM problem. In the  $H_t$  solution, we perform no SDN upgrades during the first  $t - 1$  periods, or after the  $t^{th}$  period. All the upgrades take place at the  $t^{th}$  period based on the  $ALG_t$  solution. Clearly, this is a feasible solution for PTM problem as it satisfies constraints (2) and (3). It should hold that  $J_1(H_t) \geq (T - t + 1) \cdot v(ALG_t)$ . This is because, the programmable traffic for  $H_t$  is (i) zero at each of the  $t - 1$  first time periods, (ii)  $v(ALG_t)$  at time period  $t$  and (iii) at least  $v(ALG_t)$  at each of the time periods  $t+1, t+2, \dots, T$ . The latter is due to the traffic flow demands ( $\lambda_{tf}$ ) increase with  $t$  for the same  $f$ . Then, we pick the solution  $H^*$  with the maximum programmable traffic, i.e.,  $J_1(H^*) = \max_t J_1(H_t)$ . For each  $t$ , it should hold that:

$$J_1(H^*) \geq J_1(H_t) \geq (T - t + 1) \cdot v(ALG_t) \geq (T - t + 1) \cdot \alpha \cdot v(OPT_t) \quad (6)$$

and therefore:

$$v(OPT_t) \leq \frac{1}{\alpha \cdot (T - t + 1)} J_1(H^*) \quad (7)$$

We denote by  $OPT$  an optimal solution to the PTM problem, i.e.,  $J_1(OPT) = \max_{\mathbf{x}} J_1(\mathbf{x})$ . Then, we can show that:

$$J_1(OPT) \leq \sum_{t=1}^T v(OPT_t) \quad (8)$$

This is because, when computing the value  $J_1(OPT)$ , the same budget  $B$  needs to be spread across the time periods  $t = 1, 2, \dots, T$ . However, when computing the sum  $\sum_{t=1}^T v(OPT_t)$ , new budget  $B$  is available at each time period  $t$ , without the upgrading decisions taken in  $t$  constraining the decisions in other periods. By combining equations (7) and (8), we obtain:

$$J_1(OPT) \leq \sum_{t=1}^T \frac{1}{\alpha \cdot (T - t + 1)} J_1(H^*) \quad (9)$$

and by using the definition of the  $T^{th}$  harmonic number we get:

$$J_1(OPT) \leq \frac{\log(T)}{\alpha} J_1(H^*) \quad (10)$$

or equivalently:

$$J_1(H^*) \geq \frac{\alpha}{\log(T)} J_1(OPT) \quad (11)$$

Theorems 1 and 2 analyze the complexity of the PTM problem and provide a first approximate solution. Although it is quite simple to implement, Modified-greedy provides an approximation ratio that worsens (goes to zero) as the number of periods  $T$  increases. To fill this gap, we introduce in the next section a class of (more sophisticated) algorithms with approximation ratios that are independent of  $T$  and therefore are suitable for upgrade schedules that extend over long time windows.

#### IV. TIGHT APPROXIMATIONS FOR THE PTM PROBLEM

In this section, we present tight approximation algorithms for the PTM problem by expressing the problem as the maximization of a submodular function. We begin by introducing the definition of submodular functions.

*Definition 1:* Given a finite set of elements  $\mathcal{G}$  (referred to as a ground set) a function  $H : 2^{\mathcal{G}} \rightarrow \mathbb{R}$  is submodular if for any sets  $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{G}$  and every element  $g \in \mathcal{G} \setminus \mathcal{B}$ , it holds:

$$H(\mathcal{A} \cup \{g\}) - H(\mathcal{A}) \geq H(\mathcal{B} \cup \{g\}) - H(\mathcal{B}), \quad (12)$$

i.e., the marginal value of the function when adding a new element in a set decreases as this set expands.

Let us denote the upgrade of node  $n$  at time period  $t$  by an element  $g_{tn}$  and define the ground set  $\mathcal{G}$  consisting of all elements as:

$$\mathcal{G} = (g_{tn} : t \in \mathcal{T}, n \in \mathcal{N}). \quad (13)$$

Then, every possible SDN upgrading policy can be expressed by a subset  $\mathcal{X} \subseteq \mathcal{G}$ , where the elements in  $\mathcal{X}$  correspond to the time periods and nodes that upgrades took place. Based on the above, we can write Obj 1 as a function of the set  $\mathcal{X}$ :

$$H(\mathcal{X}) = \sum_{t \in \mathcal{T}} \sum_{f \in \mathcal{F}} \lambda_{tf} 1_{\{\{\cup_{t' \leq t, n \in \mathcal{N}_f} g_{t'n}\} \cap \mathcal{X} \neq \emptyset\}}. \quad (14)$$

Then, we prove the following lemma:

*Lemma 3:* The function  $H(\mathcal{X})$  is monotone, non-decreasing and submodular.

*Proof:* Monotonicity is obvious since any new upgrade of a node cannot decrease the value of the objective function. In order to show submodularity, we observe that since the sum of submodular functions is also submodular it suffices to show that the function  $H_f(\mathcal{X}) = \sum_{t \in \mathcal{T}} \lambda_{tf} 1_{\{\{\cup_{t' \leq t, n \in \mathcal{N}_f} g_{t'n}\} \cap \mathcal{X} \neq \emptyset\}}$  for a given flow  $f$  is submodular. To this end, we consider two SDN upgrading policies  $\mathcal{A} \subseteq \mathcal{G}$  and  $\mathcal{B} \subseteq \mathcal{G}$ , where  $\mathcal{A} \subseteq \mathcal{B}$ . We also consider an element  $g_{tn} \in \mathcal{G} \setminus \mathcal{B}$  to be added to both sets. This

corresponds to upgrading node  $n$  at period  $t$ , where this upgrade was not taken place either in set  $\mathcal{A}$  or  $\mathcal{B}$ .

The marginal value for adding  $g_{tn}$  to  $\mathcal{A}$  will be zero if there exists another element  $g_{t'n'} \in \mathcal{A}$  such that  $t' \leq t$  and  $n' \in \mathcal{N}_f$ . This is because the flow  $f$  is already programmable according to policy  $\mathcal{A}$ . Else if there exists an element  $g_{t''n'} \in \mathcal{A}$  such that  $t'' > t$  and  $n' \in \mathcal{N}_f$ , then the marginal value will be  $\sum_{i=t}^{t''-1} \lambda_{if}$ . This is because the flow  $f$  now becomes programmable at time  $t$  instead of  $t''$ . Otherwise, the marginal value will be  $\sum_{i=t}^T \lambda_{if}$ .

We now consider the marginal value for adding the element  $g_{tn}$  to the set  $\mathcal{B}$ . We distinguish the following three cases:

- (i) If there exists an element  $g_{t'n'} \in \mathcal{A}$  such that  $t' \leq t$  and  $n' \in \mathcal{N}_f$  then this element will also belong to  $\mathcal{B}$ . Hence, the marginal value will be zero for both  $\mathcal{A}$  and  $\mathcal{B}$ .
- (ii) If there exists an element  $g_{t''n'} \in \mathcal{A}$  such that  $t'' > t$  and  $n' \in \mathcal{N}_f$  then this element will also belong to  $\mathcal{B}$ . We distinguish the following two subcases. (ii.a) If there exists an element  $g_{t'''n'} \in \mathcal{B} \setminus \mathcal{A}$  such that  $t''' < t''$  the marginal value for  $\mathcal{B}$  will be  $\sum_{i=t}^{t'''-1} \lambda_{if} < \sum_{i=t}^{t''-1} \lambda_{if}$ . (ii.b) Otherwise, the marginal value for  $\mathcal{B}$  will be equal to that for  $\mathcal{A}$  ( $\sum_{i=t}^{t''-1} \lambda_{if}$ ).
- (iii) In any other case, the marginal value for the set  $\mathcal{A}$  will be  $\sum_{i=t}^T \lambda_{if}$ . But, note that by definition this is the largest possible marginal value for the  $\mathcal{B}$  set as well. Hence, in all cases, we show that the marginal value is lower or equal for the set  $\mathcal{B}$  than  $\mathcal{A}$ . ■

The ground set can be partitioned into  $N$  disjoint sets,  $\mathcal{G}_1, \mathcal{G}_2, \dots, \mathcal{G}_N$ , where  $\mathcal{G}_n = \{g_{tn} : \forall t \in \mathcal{T}\}$ . Since each node can be upgraded in at most one time period, it should be  $\mathcal{X} \in \mathcal{I}_1$  where:

$$\mathcal{I}_1 = \{\mathcal{X} \subseteq \mathcal{G} : |\mathcal{X} \cap \mathcal{G}_n| \leq 1, \forall n \in \mathcal{N}\}. \quad (15)$$

Here, the pair  $(\mathcal{G}, \mathcal{I}_1)$  forms a *partition matroid constraint* [12]. Also, due to the budget constraint, it should be  $\mathcal{X} \in \mathcal{I}_2$  where:

$$\mathcal{I}_2 = \{\mathcal{X} \subseteq \mathcal{G} : \sum_{g_{tn} \in \mathcal{X}} b_{tn} \leq B\}. \quad (16)$$

Here, the pair  $(\mathcal{G}, \mathcal{I}_2)$  forms a *knapsack constraint*.

There exist various approximation algorithms for the maximization of a monotone submodular function subject to a matroid and a knapsack constraint. The algorithm with the best approximation ratio was proposed in [18]. This algorithm uses pipage rounding, a procedure which aims to convert a fractional solution of an optimization problem into an integral one, through a sequence of simple updates. It achieves a  $(1 - 1/e - \epsilon)$ -approximation ratio for any constant  $\epsilon > 0$ . Nevertheless, the value of this algorithm is mostly theoretical, since it relies on the enumeration of a number of elements which can be very large in practice.

A more practical choice is the technique presented in [19]. The idea is to reduce the knapsack constraint into a collection of partition matroids using an enumeration method. Particularly, let us denote with  $\{u_1, u_2, \dots, u_l\}$  all

**Algorithm 2:** Local search algorithm with input  $\epsilon >$ 


---

```

0
1 Set  $g_{t^*n^*} \leftarrow \operatorname{argmax}\{H(\{g_{tn}\}) \mid g_{tn} \in \mathcal{G}\}$  and
    $\mathcal{X} \leftarrow \{g_{t^*n^*}\};$ 
2 while the following operation is possible do
3   k-exchange operation: if there is a feasible  $\mathcal{X}'$ 
     such that:  $|\mathcal{X}' \setminus \mathcal{X}| \leq \frac{2}{\epsilon}$ ,  $|\mathcal{X} \setminus \mathcal{X}'| \leq \frac{2k}{\epsilon}$ ,
      $H(\mathcal{X}') \geq H(\mathcal{X})$  then
     |  $\mathcal{X} \leftarrow \mathcal{X}'$ ;
   end
end
4 Set  $x_{tn} \leftarrow 1$  if  $g_{tn} \in \mathcal{G} \forall t, n$ , otherwise zero;

```

---

the different values of the upgrade costs  $b_{tn}$ ,  $\forall t, n$  of the elements in the ground set. For example,  $l = 1$  if all elements have the same cost, but  $l = NT$  if each element has a different cost. Then, we can partition the ground set into the sets  $\mathcal{G}'_1, \mathcal{G}'_2, \dots, \mathcal{G}'_l$ , where the  $i^{\text{th}}$  set contains all elements with cost  $u_i$ . We also define  $q_i = \lfloor B/u_i \rfloor$ . In other words,  $q_i$  is the maximum number of elements in  $\mathcal{G}'_i$  that can be included in a solution without violating the knapsack constraint. Clearly, there will be at most  $m = \prod_{i=1}^l q_i$  different solutions that satisfy the knapsack constraint, one solution for each combination. For each combination  $j \in \{1, 2, \dots, m\}$  we introduce the following partition matroid:

$$\mathcal{I}_2^j = \{\mathcal{X} \subseteq \mathcal{G} : |\mathcal{X} \cap \mathcal{G}'_i| \leq q_i^j, \forall i \in \{1, 2, \dots, l\}\} \quad (17)$$

where  $q_i^j$  denotes the maximum number of elements in  $\mathcal{G}'_i$  corresponding to the  $j^{\text{th}}$  combination.

Clearly, instead of maximizing  $H(\cdot)$  function with respect to the  $\mathcal{I}_1 \cap \mathcal{I}_2$  constraint, it is equivalent to maximize  $H(\cdot)$  with respect to  $\mathcal{I}_1 \cap \mathcal{I}_2^j$ ,  $\forall j \in \{1, 2, \dots, m\}$ . The final solution will be simply the best performing of the  $m$  solutions found. In other words, with this enumeration method, we replace a knapsack constraint ( $\mathcal{I}_2$ ) with a collection of matroid constraints ( $\mathcal{I}_2^j$ ,  $j \in \{1, 2, \dots, m\}$ ). Hence, the PTM problem can be expressed as a collection of  $m$  subproblems where at each subproblem a submodular function is maximized subject to  $k = 2$  partition matroid constraints.

A local search algorithm provides a  $(1/(k + \epsilon))$ -approximation for the maximization of a submodular function subject to  $k$  partition matroid constraints [12]. This technique takes as input a parameter  $\epsilon > 0$  and maintains a solution set  $\mathcal{X}$  which is always independent in each of the  $k$  matroids. Iteratively, the algorithm tries to add at most  $2/\epsilon$  elements and delete at most  $2k/\epsilon$  elements from  $\mathcal{X}$ . If there is a local move that generates a feasible solution and improves the objective value, the algorithm repeats the local search procedure with that new solution, until no improvement is possible. The procedure is summarized in Algorithm 2.

We emphasize that the enumeration technique that we presented will be very efficient in cases that many nodes

have the same upgrade costs ( $l$  is a small number) and the budget is relatively small compared to the upgrading costs ( $q_i$  values are small numbers). Nevertheless, there may be cases in which a large number of matroid constraints need to be enumerated. In order to reduce complexity in these cases, a different enumeration method can be applied. For example, the enumeration method in [18] that approximately reduces every knapsack to a polynomial-time computable collection of matroid constraints can be used. The size of the collection can be tuned depending on how well the knapsack is approximated by the matroid constraints (cf. Lemma 3.3. in [19]).

The pipage rounding and the local search algorithms provide approximation ratios that are independent of the number of periods  $T$ . Specifically, the following theorem holds:

*Theorem 3:* There exists a  $(1 - 1/e - \epsilon)$ -approximation algorithm and an  $(1/(2 + \epsilon))$ -approximation algorithm to the PTM problem for any  $\epsilon > 0$ .

## V. OPTIMIZING THE TE FLEXIBILITY

In this section, we study the SDN upgrading problem when the ISP's goal is to maximize the TE flexibility. This is achieved through the availability of alternative paths that can be dynamically activated in such hybrid SDN networks. Namely, the ISP can use these paths to avoid congestion in cases that certain links are temporarily overloaded or failed. For example, as explained in Figure 1, when node 1 is upgraded the flow can be dynamically routed towards the alternative path 1; and if node 4 is also upgraded, then the ISP can also activate alternative path 2, if needed. Note that TE flexibility also improves network operation resilience when the network is under stress (due to attacks or outages) because it provides alternative paths to reroute the flows.

To formulate this as an optimization problem, we introduce a set  $\mathcal{P}_f$  for each flow  $f$ . This set includes alternative paths that flow  $f$  can follow to reach its destination. For example, an ISP can analyze historical network data to predict which paths will be underutilized, and hence they can be dynamically activated to carry flow  $f$  when its shortest path becomes congested. For each alternative path  $p \in \mathcal{P}_f$ , we define a group of nodes  $s_{pf}$  which *all* need to be upgraded to SDN in order for path  $p$  to become available to route flow  $f$ . For example, in Figure 1 we have  $s_{\text{alternative\_path\_1}, f} = \{1\}$  and  $s_{\text{alternative\_path\_2}, f} = \{1, 4\}$ . A generic approach to compute these groups was presented in [20], [21].

There will be a TE benefit  $w_{pft} \geq 0$  if alternative path  $p \in \mathcal{P}_f$  is available for routing flow  $f$  at time period  $t$ . A similar modeling approach of TE benefit has been considered in [20], [21]. This benefit will be higher for flows that carry large volumes of traffic and paths with sufficient spare capacity to carry these volumes. For a given upgrading policy  $\mathbf{x}$ , the total TE benefit (or flexibility) can be expressed



as follows:

$$J_2(\mathbf{x}) = \sum_{t \in \mathcal{T}} \sum_{f \in \mathcal{F}} \sum_{p \in \mathcal{P}_f} w_{pft} 1_{\{\prod_{n \in s_{pf}} [\sum_{t' \leq t} x_{t'n}] > 0\}}. \quad (18)$$

Here, the benefit  $w_{pft}$  is earned when *all* nodes in the group  $s_{pf}$  have been upgraded to SDN by time  $t$ . In the special case that  $w_{pft} = 1 \forall p, f, t$ , the total TE benefit matches the total number of alternative paths that are enabled by SDN nodes.

The objective of the ISP is to find the SDN upgrading policy that maximizes the TE flexibility (*TEFM problem*):

$$\begin{aligned} \text{Obj2 :} \quad & \max_{\mathbf{x}} \quad J_2(\mathbf{x}) \\ \text{s.t. constraints :} \quad & (2), (3), (5) \end{aligned}$$

Obj2 is more complex than Obj1, since it depends on the exact set of SDN nodes that each flow traverses, rather than on whether the flow traverses at least one of them. In this section, we present an initial approach to optimize this objective. Our idea is to express the TEFM problem as the maximization of a set function with bounded supermodular degree [13]. We begin with the following definitions.

**Definition 2:** The supermodular degree of an element  $g \in \mathcal{G}$  by a function  $H(\cdot)$  is defined as the cardinality of the set  $\mathcal{D}_H^+(g) := \{g' \in \mathcal{G} : \exists \mathcal{X} \subseteq \mathcal{G} \text{ for which } H(\mathcal{X} \cup \{g'\}) \cup \{g\} - H(\mathcal{X} \cup \{g'\}) > H(\mathcal{X} \cup \{g\}) - H(\mathcal{X})\}$ . In other words, the set  $\mathcal{D}_H^+(g)$  contains all elements  $g'$  the existence of which in a set might increase the marginal value of element  $g$ .

**Definition 3:** The supermodular degree of a function  $H(\cdot)$ , denoted by  $D_H^+$ , is simply the maximum supermodular degree of any element  $g \in \mathcal{G}$ . Formally,  $D_H^+ = \max_{g \in \mathcal{G}} |\mathcal{D}_H^+(g)|$ .

It is not hard to express  $J_2$  as a function with bounded supermodular degree ( $D_{J_2}^+$ ). In fact, all the nodes that are included in the same group depend on each other, in the sense that upgrading only one of them yields no TE benefit, but when they are all upgraded a benefit is earned ( $w_{pft}$  for group  $s_{pf}$ ). For example, in Figure 1, the marginal value for upgrading node 4 is zero if node 1 is not already upgraded, since the only available path for the flow would be still the shortest path. But, if node 1 is already upgraded, then the marginal value for upgrading node 4 becomes positive (since routing over alternative path 2 now becomes possible). Hence, the supermodular degree of function  $J_2$  is 1 in this example. For a single time period, the supermodular degree of  $J_2$  is simply the maximum number of nodes that share the same group with any other node. Intuitively, the more the path  $p$  diverges from the initial shortest path of the flow  $f$  the more node upgrades are needed to enable such re-routing, and hence the larger the  $s_{pf}$  group becomes. The exact number of such nodes depends on how the shortest and alternative paths overlap. In general, it holds  $0 \leq D_{J_2}^+ \leq NT - 1$ .

A variant of the greedy algorithm has been proposed for maximizing any function  $H$  with bounded supermodular degree  $D_H^+$ . We call this the Super-greedy algorithm (see

---

**Algorithm 3:** Super-greedy algorithm

---

```

1  $\mathcal{X} \leftarrow \emptyset$ ;
2 while there exists an element  $g$  such that  $\mathcal{X} \cup \{g\}$  is
  a feasible solution do
3   Let  $g^* \in \mathcal{G} \setminus \mathcal{X}$  and  $\widehat{\mathcal{D}}_H^+(g^*) \subseteq \mathcal{D}_H^+(g^*)$  be a pair
    of an element and a set such that:
    (i)  $\mathcal{X}' := \mathcal{X} \cup \{g^*\} \cup \widehat{\mathcal{D}}_H^+(g^*)$  is a feasible solution,
    and
    (ii) it maximizes  $H(\mathcal{X}') - H(\mathcal{X})$ ;
4    $\mathcal{X} \leftarrow \mathcal{X}'$ ;
end
5 Set  $x_{tn} \leftarrow 1$  iff  $g_{tn} \in \mathcal{X}$ , otherwise 0;
```

---

Algorithm 3). This algorithm starts with an empty solution set  $\mathcal{X}$  (line 1) and iteratively augments subsets of elements to it (lines 2-4). At each iteration, it picks an element  $g^*$  and a subset of those elements that increase the marginal value of  $g^*$ , i.e., a subset of the set  $\mathcal{D}_H^+(g^*)$ . The above choice is made greedily so that the highest marginal benefit is earned. The procedure ends when there are no more elements to augment.

The work in [13] has shown that Super-greedy achieves an approximation ratio for the problem of maximizing a function with bounded supermodular degree. Nevertheless, a necessary condition for this result to hold is that the constraints of the problem form a  $k$ -extendible system, which is a class of constraints that capture the case of  $k$  matroid constraints, but not the case of knapsack constraints. For the special case that all the upgrading costs are equal, the constraints (2), (3) can be expressed as  $k = 2$  matroid constraints. Therefore, we obtain:

**Proposition 1:** Super-greedy achieves an  $(1/(2(D_{J_2}^+ + 1) + 1))$ -approximation ratio to the TEFM problem for the special case of uniform upgrading costs.

## VI. MINIMIZING THE UPGRADE COSTS

So far, we focused on the investment of a given economic budget  $B$  in SDN upgrades, aiming to optimize a performance objective (Obj1 or Obj2). In certain cases, however, an ISP may be more concerned about meeting *specific performance targets* rather than spending a *specific budget*. These performance targets may span one or multiple time periods. For example, an ISP may prefer to upgrade its network in a way that directly makes 30% or more of its traffic programmable, while another ISP may opt for 10% the first year, 20% the second year, etc. Similarly, the targets can be with respect to Obj2. This gives rise to a different, yet equally important, version of the SDN upgrading problem. The objective in this “dual” version is to minimize the cost spent in upgrades while ensuring specific levels of performance.

For the sake of completeness, in this section, we study such a minimization version of the SDN upgrading problem where the targets are with respect to programmable traffic.

In each time period  $t \in \mathcal{T}$  the upgrading policy must ensure that at least  $P_t \in [0, 1]$  portion of the traffic is made programmable, where typically  $P_{t'} > P_t$  for  $t' > t$ . Formally, the following constraints must be satisfied:

$$\frac{\sum_{f \in \mathcal{F}} \lambda_{tf} 1_{\{\sum_{n \in \mathcal{N}_f} \sum_{t' \leq t} x_{t'n} \geq 1\}}}{\sum_{f \in \mathcal{F}} \lambda_{tf}} \geq P_t, \quad \forall t \in \mathcal{T}. \quad (19)$$

The optimal upgrading policy will minimize the total cost spent in upgrades (*UCM problem*):

$$\begin{aligned} \text{Obj3:} \quad & \min_{\mathbf{x}} \sum_{t \in \mathcal{T}} \sum_{n \in \mathcal{N}} x_{tn} b_{tn} \\ \text{s.t. constraints:} \quad & (3), (5), (19) \end{aligned}$$

Subsequently, we investigate the complexity of the UCM problem. We focus on the special, yet practical, case of a single time period ( $T = 1$ ), i.e., one-step upgrade. Even for this case, we show that the problem is NP-Hard and present an approximation algorithm.

*Theorem 4:* For the special case of  $T = 1$  time period, the UCM problem is NP-Hard, but there exists a polynomial-time  $(4/3 + \epsilon)H(\Delta)$ -approximation algorithm, where  $\epsilon$  is any positive constant,  $\Delta$  is the maximum number of flows that traverse a node and  $H$  is the harmonic number, i.e.,  $H(q) = \sum_{k=1}^q 1/k$ .

To prove the above theorem, we show the equivalence to the following variant of the cover problem [22].

*Definition 4: Generalized Partial Cover (GPC):* Given a set of elements  $\mathcal{E} = \{E_1, E_2, \dots, E_l\}$  with associated weights  $\{w_i\}_{i=1}^l$  and a collection of sets  $\mathcal{S} = \{S_1, S_2, \dots, S_m\}$  defined over  $\mathcal{E}$  with associated costs  $\{c_i\}_{i=1}^m$ , the goal is to find a collection of sets  $\mathcal{S}' \subseteq \mathcal{S}$  such that the total weight of the elements covered by  $\mathcal{S}'$  is at least  $L$ , a specified lower bound, and the total cost of sets in  $\mathcal{S}'$  is minimized.

Specifically, we prove the following lemma.

*Lemma 4:* UCM for  $T = 1$  and GPC are equivalent problems.

*Proof:* The proof is similar to that of Lemma 1. Given an instance of the GPC problem, we create a separate node for each set (with upgrading cost equal to the cost of the set) and a separate flow for each element (with rate equal to the weight of the element). If there exists a set of nodes, such that by upgrading the nodes in this set to SDN makes  $P_t = L$  or more traffic programmable, then there will also be a collection of sets that cover elements with total weight  $P_t$  or more, having the same total cost. ■

Building on Lemma 4, we can use the approximation algorithms that have been proposed for GPC in order to solve UCM for a single time period  $t$ . Below, we explain one of these algorithms, proposed in [22].

First, we introduce the auxiliary variable  $z_{tf} \in \{0, 1\}$  which indicates whether flow  $f$  is *not* programmable in period  $t$ . The respective vector of variables for period  $t$  will be  $\mathbf{z}_t = (z_{tf} : f \in \mathcal{F})$ . Similarly, we define  $\mathbf{x}_t = (x_{tn} : n \in \mathcal{N})$ .

Second, we define the *Lagrangian Relaxation* of the UCM problem (referred to as *LR-UCM<sub>t</sub>*). This is realized by

dualizing the constraint in (19) and lifting it to the objective function multiplied by  $\mu \geq 0$ :

$$\begin{aligned} \min_{\mathbf{x}_t, \mathbf{z}_t} \quad & \sum_{n \in \mathcal{N}} x_{tn} b_{tn} + \mu \left( \sum_{f \in \mathcal{F}} \lambda_{tf} z_{tf} - (1 - P_t) \sum_{f \in \mathcal{F}} \lambda_{tf} \right) \\ \text{s.t.} \quad & \sum_{n \in \mathcal{N}_f} x_{tn} + z_{tf} \geq 1, \quad \forall f \in \mathcal{F}, \end{aligned} \quad (20)$$

$$x_{tn}, z_{tf} \in \{0, 1\}, \quad \forall n \in \mathcal{N}, f \in \mathcal{F}. \quad (21)$$

In the above formulation, there is no restriction to make any flow programmable; however, if the nodes we upgrade leave a flow  $f$  un-programmable ( $z_{tf} = 1$ ), we incur a penalty of  $\mu \lambda_{tf}$ .

Third, we use a simple greedy scheme to obtain an approximate solution to the LR-UCM<sub>t</sub> problem [22]. That is, starting with an empty solution ( $\mathbf{x}_t = 0, \mathbf{z}_t = 0$ ), we iteratively set the value of a variable ( $x_{tn}$  for some  $n$  or  $z_{tf}$  for some  $f$ ) from zero to one such that the *average gain* is maximized. When a variable  $x_{tn}$  is set to one, the average gain will be the number of flows that become programmable and for which  $z_{tf} = 0$  over  $b_{tn}$ . However, when a variable  $z_{tf}$  is set to one, the average gain will be 1 over  $\mu \lambda_{tf} H(\Delta)$ . Note, here, that the cost in the denominator is inflated by a factor  $H(\Delta)$ . This greedy scheme will end when every flow  $f$  has either become programmable or  $z_{tf} = 1$ .

Fourth, we use as a “black-box” the solution to LR-UCM<sub>t</sub> to solve the UCM problem for period  $t$ . Specifically, we conduct a *binary search* over the possible values of  $\mu$  which consists of a polynomially-bounded number of calls to the greedy scheme for LR-UCM<sub>t</sub>. The binary search is over the following interval:

$$\mu \in \left[ 0, \frac{2 \sum_{n \in \mathcal{N}} b_{tn}}{\min_{f \in \mathcal{F}} \lambda_{tf}} \right] \quad (22)$$

The result of the binary search will be two values  $\mu_1 \geq \mu_2$  in the above interval that satisfies:

- 1)  $\mu_1 - \mu_2 \leq \epsilon \min_n b_{tn}$ , where  $\epsilon$  is any given positive constant,
- 2) the programmable traffic corresponding to the solution for  $\mu_1$  is at least  $P_t$ , and at the same time, the programmable traffic for  $\mu_2$  is at most  $P_t$ .

The solution for  $\mu_1$ , denoted by  $\mathbf{x}_t^1$ , will always be feasible, while the solution for  $\mu_2$ , denoted by  $\mathbf{x}_t^2$ , may not be feasible.

The fifth and last step is to create an additional feasible solution  $\mathbf{x}_t^3$ , by augmenting  $\mathbf{x}_t^2$  with a subset of nodes upgraded in  $\mathbf{x}_t^1$ . We do this greedily by upgrading the node with the highest ratio of traffic that becomes programmable over upgrading cost until the  $P_t$  target is met. Finally, among the two feasible solutions  $\mathbf{x}_t^1$  and  $\mathbf{x}_t^3$ , the one with the lowest upgrading cost is picked. This solution will be within a factor of  $(4/3 + \epsilon)H(\Delta)$  from optimal [22].

## VII. DATASET-DRIVEN EVALUATION

In this section, we evaluate the performance of the proposed algorithms using real-world network topologies

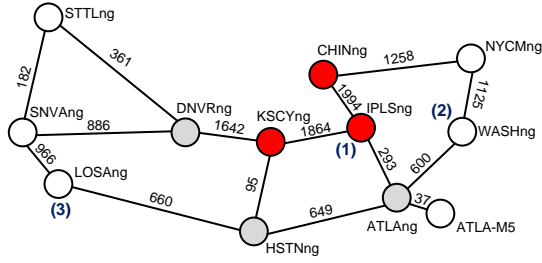


Fig. 3. The Abilene network. SDN upgraded nodes returned by DEG (highlighted in light gray), VOL (highlighted in red) and Local Search (indexed by (1), (2), (3)) algorithms for  $T = 1$  and  $B = \$300K$ . Node labels correspond to router names and link labels correspond to aggregate traffic volumes (in Mbps).

and traffic matrices. Overall, we find that our approach can increase by 54% the amount of programmable traffic compared to state-of-the-art methods, especially in practical scenarios where the network is upgraded in a time window of four or five years. In general, the ISP acquires more benefits by spreading the upgrades over many instead of one year. Nevertheless, this strategy can be detrimental when the SDN costs are relatively stable over time (up to 20% drop per year). We also find that by optimizing the objective of programmable traffic maximization, benefits are also realized for the objective of TE flexibility maximization (and vice versa). However, there will be a performance loss (up to a factor of 2), since each algorithm favors one objective over the other.

We have implemented the following eight algorithms:

- 1) **DEG** [33]: This scheme upgrades the nodes with the highest degrees (number of incoming and outgoing adjacent links) in the topology graph, until budget  $B$  is spent. All the upgrades take place at the first time period.
- 2) **VOL** [11], [33]: This scheme upgrades the nodes with the highest traffic volume that traverses them, until budget  $B$  is spent. All the upgrades take place at the first time period.
- 3) **Modified-greedy**: The proposed scheme in Algorithm 1 extended for many time periods.
- 4) **Local search**: The proposed scheme in Algorithm 2 for  $\epsilon = 2$  that can spread upgrades over many time periods.
- 5) **Super greedy**: The proposed scheme in Algorithm 3 that maximizes TE flexibility.
- 6) **MUCPF** [27]: This scheme upgrades the node that covers the maximum number of flows until the minimum programmable traffic target is met. All the upgrades take place at the first time period.
- 7) **Highest ratio** [28]: This scheme upgrades the node with the highest ratio of traffic volume that traverses it over upgrading cost, until the minimum programmable traffic target is met. All the upgrades take place at the first time period.
- 8) **Binary search**: The proposed scheme in Section VI

for  $\epsilon = 0.1$  that minimizes upgrading cost. All the upgrades take place at the first time period.

The first four algorithms above will be compared with respect to Obj1 subject to a specific budget  $B$ . The fifth algorithm will be evaluated with respect to Obj2 subject to the same budget. Finally, the last three algorithms will be compared with respect to Obj3 subject to a specific performance target  $P_t$ .

The main part of the evaluation is carried out using the Abilene dataset [14] which is obtained from an educational backbone network in North America. This network consists of 12 nodes and 30 directed links as depicted in Figure 3. The dataset records the traffic matrix, i.e., the data transmitted between every pair of nodes, every 5 minutes for an overall period of six months. We use the traffic matrix at 8:00 pm on the first day to set the rates of the respective 144 flows. The aggregate rate is found to be 5.46 Gbps. These rates correspond to the  $\lambda_{tf}$  values for period  $t = 1$ . We increase the rates in subsequent periods (years) by 22% ( $\lambda_{tf} = \lambda_{t-1f} \cdot 122\%$ ) [16]. The dataset also records the OSPF weights of all the links, which allows us to find the shortest path between every pair of nodes ( $\mathcal{N}_f$  sets).

We emphasize that we focus on this specific subset of the dataset because it represents a peak time period when SDN is more important. Moreover, this dataset is publicly available online, whereas data from most ISPs is proprietary. The evaluation code we wrote is publicly available online [35]. We believe that the reproducibility of the results will encourage future experimentation with SDN algorithms.

We start with Obj1 and examine how the proposed algorithms compare with the state-of-the-art methods. Since the latter neglect the timing issue, and in order to ensure a fair comparison, we begin our investigation with  $T = 1$  time period, i.e., all upgrades take place within one year. As a canonical scenario, we set the cost of upgrading a node to SDN to \$100K [7] ( $b_{tn}$  values for  $t = 1$ ), and we vary the budget  $B$  from \$100K to \$1M (Figure 4(a)). We observe that as the budget increases, the volume of programmable traffic increases for all algorithms. This is because more nodes are upgraded to SDN which creates more opportunities for the flows to traverse SDN nodes. There exists a saturation point ( $B = \$900K$ ), after which no significant changes are noticed. *The proposed algorithms (Modified greedy and Local search) achieve up to 54% more programmable traffic than their counterparts.*

To better understand how the algorithms work and why we obtain the above gains, we depict the upgrading decisions of the algorithms in Figure 3. While DEG upgrades the three nodes with the highest degrees (ATLAng, HSTNng and DNVRng), VOL picks the IPLSng, KSCYng and CHINng nodes which cover the most heavy-loaded links. Local search upgrades first the IPLSng node similar to VOL. However, the next decisions are different as most of the traffic that KSCYng and CHINng cover is already made programmable by the upgrade of IPLSng. Therefore, two different nodes will be picked (WASHng and LOSAng). We note that the

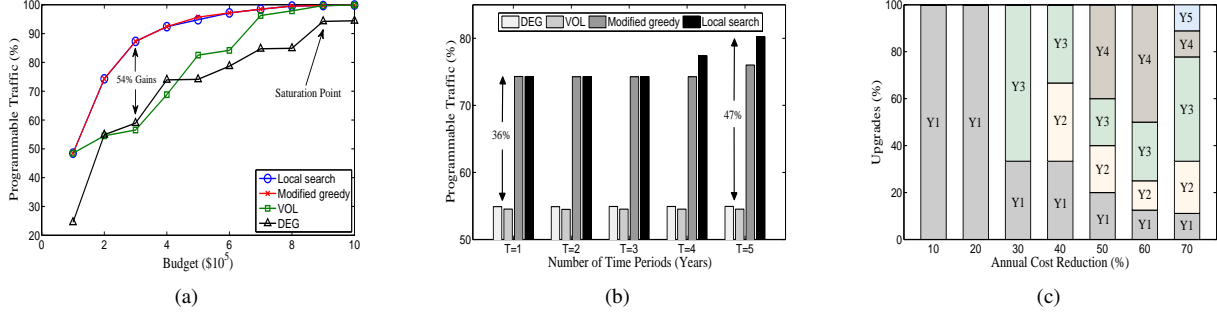


Fig. 4. The programmable traffic achieved by DEG, VOL, Modified greedy and Local search algorithms as a function of (a) the budget  $B$  and (b) the number of time periods  $T$ . (c) The distribution of upgrades across years (Y1, Y2, Y3, Y4, Y5) for different cost reduction rates.

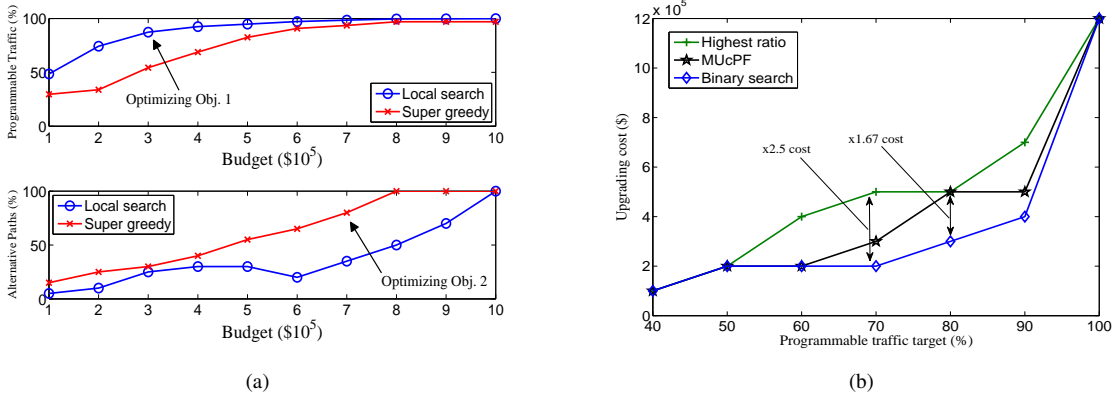


Fig. 5. (a) Impact of optimizing different objectives for  $T = 1$ . (b) Upgrading cost (Obj3) for different minimum levels of programmable traffic.

Modified Greedy algorithm will return the same upgrading policy with Local search, which happens to be optimal in this case.

We note that since the Abilene network is fairly small ( $N = 12$  nodes) and for  $T = 1$  time period, we can compute the optimal solution in reasonable time by using exhaustive search methods. That is by enumerating all the  $2^{12} = 4,096$  possible solutions and then picking the solution that yields the largest programmable traffic. By carrying out this process, we observed that Modified greedy and Local search algorithms perform very close to the optimal (less than 1% for the scenarios in Figure 4(a)). However, we are not able to apply exhaustive search methods to find the optimal solution in scenarios of larger  $T$  or larger networks.

We then explore the impact of the number of time periods  $T$  in Figure 4(b). Here, we keep  $B = \$200K$  constant, but we vary  $T$  within 1 to 5 years. To capture technology maturity, we decrease the SDN upgrading costs by 40% per year, i.e.,  $b_{tn} = b_{t-1n} - b_{t-1n} \cdot 40\%$ . For  $T = 1$ , the results match those in Figure 4(a). For  $T > 1$ , additional benefits can be acquired by postponing some of the upgrades after the first year when the costs will be lower. *Local search algorithm intelligently spreads the upgrades across different years to achieve the best performance among the four*

*algorithms*. The benefits over the state-of-the-art methods are up to 47%, and 5.5% over Modified greedy for  $T = 5$ .

In Figure 4(c), we take a closer look into the distribution of upgrades over years when the Local search algorithm is used. We evaluate various scenarios which differ into the annual decrease rate of the upgrading costs. We find that *for relatively low rates of cost decrease (up to 20%), all the upgrades should take place within the first year. But, after this point, it is more beneficial to postpone some of the upgrades in the future*. The distribution of upgrades over years becomes more diverse as the rate of cost decrease increases.

Then, we explore the interplay between traffic programmability (Obj1) and TE flexibility benefits (Obj2). As we showed in previous figures, Local Search is in practice a very efficient algorithm for maximizing programmable traffic. But, a large volume of programmable traffic cannot guarantee by itself a large number of alternative routing paths (and vice versa). Therefore, it is questionable how well an algorithm that optimizes one of the two objectives will perform with respect to the other objective. Figure 5(a) aims to shed light on this issue by comparing the performance of the Local search algorithm (which optimizes programmable traffic) and Super-greedy (which optimizes TE flexibility).

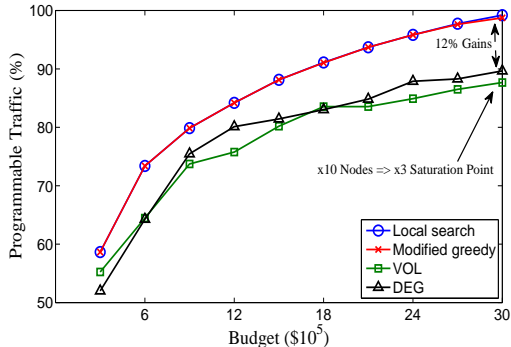


Fig. 6. Programmable traffic for the Deltacom network [15].

Here, to model the TE benefits, we focus on the 10 flows with the highest rate, for which TE is most important. Then, we consider as alternative paths the second and third shortest path for each flow that does not overlap with the shortest path ( $\mathcal{P}_f$  sets). We find that *by optimizing one of the objectives, benefits are also realized for the other objective. However, there will be a performance loss (up to a factor of 2), since each algorithm favors one objective over the other.*

We also present evaluation results for Obj3 to examine how the proposed algorithm (Binary search) compares with the state-of-the-art methods (MUcPF and Highest ratio). We carry this out for a single time period  $t$  and different values of the programmable traffic target ( $P_t$ ). The results are depicted in Figure 5(b). As expected, the cost spent on upgrades increases with  $P_t$  for all the algorithms. Interestingly, *the proposed algorithm requires significantly lower cost than its counterparts; up to 2.5 and 1.67 times lower than Highest ratio and MUcPF respectively.*

Although in our evaluation we used a real network topology and traffic matrices, it would be also interesting to study the results in larger networks. Towards this goal, we use the topology of the Deltacom backbone network in North America, which consists of 113 nodes and 161 links, and it is publicly available online in [15]. Since there is no available information about the traffic, we generate this artificially. Particularly, we create  $F = 1,000$  flows, by picking uniformly at random origin-destination pairs. We compute the shortest paths based on the hop count length, and we set the flow rate to be disproportional to it (following the gravity model [36]). In Figure 6, we repeat the experiment presented in Figure 4(a), but for this larger network. We find that *the proposed algorithms perform up to 12% better than their counterparts. The saturation point is found to be  $B = \$3M$ , about three times larger than in the small network.* We attribute this difference to the larger number of nodes (10x) in the Deltacom network and the topological characteristics, as the Deltacom has a higher link density which enables SDN nodes to cover more flows. The running times of the algorithms are typically on the scale of minutes for the small network and hours for the large network. For the Local search algorithm specifically,

the running time will be less than 10 minutes even if we consider all the possible 12656 ( $113 \times 112$ ) flows in the large network. These are acceptable running times in practice since the problem has to be solved offline by the ISP.

## VIII. RELATED WORK

In this section, we present related work on SDN upgrading. We also put our work in perspective in Table I that lists the main papers that tackle this problem, categorized based on their contributions.

**Hybrid SDN.** Incremental deployment of new protocols and architectures is an operational paradigm shift [5], and SDN is no exception to that. Namely, several techno-economic factors make ISPs reluctant to proceed with immediate full-scale SDN deployment. This renders hybrid SDN networks an imperative intermediate step [6]. Such systems are nowadays possible due to hybrid routers [23], yet their deployment is not without challenges. For example, the co-existence of multiple control planes poses risks for fault-free routing, and specific measures should be taken to avoid this, e.g., see [24].

One of the key traffic engineering goals in these hybrid networks is to use the SDN routers so as to minimize the load of congested links, e.g., see seminal work [9], and [25], [26]. For a given set of upgraded nodes, this can be expressed as an LP problem. However, upgrading decisions are more challenging as they yield intractable problem formulations.

**SDN upgrading problem.** References [27] and [28] proposed meaningful heuristics for the programmable traffic maximization and upgrading cost minimization objective respectively. Reference [29] designed a cover-based approximation algorithm for the programmable traffic maximization objective. A distinctive feature of the latter work is that the SDN devices do not replace the legacy ones, but are deployed in addition to them. In another front, the work in [30] studied the objective of maximizing the minimum number of loop-free routing paths enabled by the SDN nodes. This TE-based objective can have a positive impact on security by eliminating the routing bottlenecks likely to be utilized by an adversary. Another approach that deploys SDN nodes in a way that partitions the network into sub-domains so as to achieve TE capabilities comparable to full SDN deployment, was proposed in [31]. While the above works considered similar objectives to our work, they did not provide approximation bounds (with the exception of [29]), nor they studied the timing aspect.

**Joint SDN upgrading & routing problem.** Another series of works follow a joint SDN upgrading and routing optimization approach, i.e., the flow routing and node upgrading decisions are jointly made. The pioneer work in [11] proposed a joint scheme that deploys SDN nodes and re-routes all flows through at least one of them (hence all the traffic is made programmable), with the goal of minimizing the total path stretch. Another joint scheme that assigns to each legacy node an SDN node, able to dynamically receive traffic and perform failover, was proposed in [32].



TABLE I  
RELATED WORKS ON SDN UPGRADING.

Reference	Objective				Joint routing & upgrading	Timing aspect
	Max programmable traffic	Max TE gains	Min upgrade cost	Min path stretch		
[27]	✓	✓	✓	✗	✗	✗
[28]	✓	✗	✓	✗	✗	✗
[29]	✓	✗	✗	✗	✗	✗
[30],[31]	✗	✓	✗	✗	✗	✗
[11]	✗	✗	✗	✓	✓	✗
[32]	✗	✗	✓	✗	✓	✗
[33],[34]	✗	✓	✗	✗	✓	✗
[20],[21]	✗	✓	✗	✗	✗	✓
<b>This work</b>	✓	✓	✓	✗	✗	✓

Here, the objective was to minimize the total number of upgraded nodes subject to certain assignment restrictions. Two more joint schemes have been proposed in the literature; a heuristic that minimizes the maximum link usage [33] and a randomized-based approximation algorithm that maximizes network throughput [34].

We note that although the above joint approaches are meaningful in certain cases, in general, the ISP will solve the SDN upgrading and routing problems in *different timescales*. A typical SDN upgrade window may span several years. On the contrary, the routing problem should be solved in a shorter time scale to respond to dynamic network conditions. Therefore, in this work, we tackle the SDN upgrading problem independently from routing, using long-term traffic predictions (as in [27], [28], [29], [30], [31], [20], [21]).

**Timing aspect.** Besides, one of our main focal points is the impact of upgrade timing. This is a very crucial and practical issue in hybrid SDNs given that (i) new technology costs reduce rapidly [37], (ii) the out-of-phase life-cycles of the legacy devices render cost-prohibitive massive replacements, and (iii) the practical, technical and security limitations render impossible one-time upgrades. Prior works as those above do not focus on these aspects. On the contrary, few prior interesting works [20], [21] studied gradual upgrades, yet, they do not provide tight bounds, nor they analyze the impact of equipment cost reduction. Clearly, the necessary consideration of time dimension increases further the problem's complexity. To cope with this issue, we carefully employ state-of-the-art algorithms such as [12], which can provide good-guarantees even for large networks.

**Alternatives to hybrid SDN.** Prior work also tries to achieve SDN-like flexible path enforcement with legacy networks. Fibbing [38] injects fake nodes and links into the underlying link state routing protocol to achieve some level of load balancing and TE, but its forwarding rule matching is limited to destination-based, and its expressivity is thus confined to the expressiveness of IP routing. Besides, with injected "lies", Fibbing could lead to debugging issues and incorrect operation. DEFO [40] leverages segment routing to control routing paths for carrier-grade traffic engineering but shares some similar limitations as in Fibbing. Besides, its constraint programming based middle-point selection largely focuses on static traffic matrices, while our proposed TE

module can load balance dynamic traffic demands.

## IX. CONCLUSION

In this paper, we studied the migration to SDN of high-end ISP core networks. To this end, we introduced a model of gradual SDN upgrades general enough to capture different ISP migration strategies, costs, and objectives. An ISP can apply our methodology to optimally decide which nodes to upgrade over a period that may span several years. We focused on two popular objectives of ISPs, namely, (i) the maximization of the programmable traffic that traverses at least one of the SDN enabled nodes, and (ii) the maximization of the traffic engineering flexibility, i.e., the increase in the number of alternative paths available to flows achieved by SDN upgrades. For these two objectives we characterized the complexity of the problem and we proposed algorithms to achieve the optimal upgrading schedule. We also studied the dual problem of minimizing the cost of SDN upgrades to ensure specific performance goals. Using two real-world network topologies and traffic matrices, we differentiated situations in which upgrades should be spread over many instead of one step and explored the interplay between the different objectives. Our results showed that for the first two objectives, the performance of our algorithms showed up to 54% gains over state-of-the-art methods and for the dual problem our proposed algorithm achieved up to 2.5 times lower cost to ensure performance goals over state-of-the-art methods.

## REFERENCES

- [1] K. Poularakis, G. Iosifidis, G. Smaragdakis, L. Tassiulas, "One Step at a Time: Optimizing SDN Upgrades in ISP Networks", *IEEE Infocom*, 2017.
- [2] W. Xia, Y. Wen, C. Foh, D. Niyato, "A Survey on Software-defined Networking", *IEEE Commun. Surveys & Tutor.*, vol. 17, no. 1, 2015.
- [3] J. Moy, "OSPF Version 2", RFC 2328, April 1998.
- [4] D.O. Awduche, "MPLS and traffic engineering in IP networks", *IEEE Communications Magazine*, vol. 37, no. 12, pp. 42-49, 1999.
- [5] M. K. Mukerjee, D. Han, S. Seshan, P. Steenkiste, "Understanding Tradeoffs in Incremental Deployment of New Network Architectures", in *Proc. of ACM CoNEXT*, 2013.
- [6] S. Vissicchio, L. Vanberer, O. Bonaventure, "Opportunities and Research Challenges of Hybrid Software Defined Networks", *ACM CCR*, vol. 44, no. 2, 2014.
- [7] Reuters Technology News, "AT&T to buy Cisco Core Routers for Network Upgrade", [Online:] [www.reuters.com](http://www.reuters.com)
- [8] Z. Cao, M. Kodialam, T. V. Lakshman, "Traffic Steering in Software Defined Networks: Planning and Online Routing", *ACM DCC*, 2014.

- [9] S. Agarwal, M. Kodialam, T.V. Lakshman, "Traffic Engineering in Software Defined Networks", *IEEE Infocom*, 2013.
- [10] Light Reading Portal, "NEC Slashes OpenFlow SDN Controller Pricing", [Online]: <http://www.lightreading.com/carrier-sdn/sdn-technology/nec-slashes-openflow-sdn-controller-pricing/d/d-id/711391>, 2014.
- [11] D. Levin, M. Canini, S. Schmid, F. Schaffert, A. Feldmann, "Panopticon: Reaping the Benefits of Incremental SDN Deployment in Enterprise Networks", in *Proc. of USENIX ATC*, 2014.
- [12] J. Lee, M. Sviridenko, J. Vondrak, "Submodular Maximization over Multiple Matroids via Generalized Exchange Properties", *Math. of Operations Research*, vol. 35, no. 4, 2010.
- [13] M. Feldman, R. Izsak, "Constrained Monotone Function Maximization and the Supermodular Degree", in *Proc. of APPROX/RANDOM*, 2014.
- [14] Abilene dataset, <http://www.cs.utexas.edu/~yzhang/research/AbileneTM>
- [15] "The Internet Topology Zoo", <http://www.topology-zoo.org/dataset.html>
- [16] Cisco White Paper, "VNI: Forecast and Methodology, 2015-2020", 2016.
- [17] S. Khuller, A. Moss, J. Naor, "The Budgeted Coverage Problem", *Information Processing Letters*, vol. 70, no. 1, 1999.
- [18] C. Chekuri, J. Vondrák, R. Zenklusen, "Dependent Randomized Rounding via Exchange Properties of Combinatorial Structures", in *Proc. of FOCS*, 2010.
- [19] A. Gupta, V. Nagarajan, R. Ravi, "Robust and MaxMin Optimization under Matroid and Knapsack Uncertainty Sets", *ACM Transactions on Algorithms*, vol. 12, no. 1, 2016.
- [20] M. Caria, A. Jukan, M. Hoffmann, "A Performance Study of Network Migration to SDN-enabled Traffic Engineering", *IEEE Globecom*, 2013.
- [21] T. Das, M. Caria, A. Jukan, M. Hoffmann, "A Techno-economic Analysis of Network Migration to Software-Defined Networking", *arXiv. 1310.0216v1*, 2013.
- [22] J. Konemann, O. Parekh, D. Segev, "A Unified Approach to Approximating Partial Covering Problems", *Algorithmica*, vol. 59, no. 4, pp. 489-509, 2011.
- [23] Open Networking Foundation, "OpenFlow Switch Specification", March 2015.
- [24] S. Vissicchio, L. Cittadini, O. Bonaventure, G. G. Xie, L. Vanberer, "On the Co-Existence of Distributed and Centralized Routing Control Planes", in *Proc. of IEEE Infocom*, 2015.
- [25] Y. Guo, Z. Wang, X. Yin, X. Shi, J. Wu, "Traffic Engineering in SDN/OSPF Hybrid Network", in *Proc. of IEEE ICNP*, 2014.
- [26] J. He, and W. Song, "Achieving Near-Optimal Traffic Engineering in Hybrid Software Defined Networks", in *Proc. of IFIP Networking*, 2015.
- [27] E. H.-K. Wu, B. Kar, and Y.-D. Lin, "The Budgeted Maximum Coverage Problem in Partially Deployed Software Defined Networks", *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 394-406, 2016.
- [28] X. Jia, Y. Jiang, Z. Guo, "Incremental Switch Deployment for Hybrid Software-Defined Networks", *IEEE LCN*, 2016.
- [29] H. Xu, X.-Y. Li, L. Huang, H. Deng, H. Huang, and H. Wang, "Incremental Deployment and Throughput Maximization Routing for a Hybrid sdn", *IEEE/ACM Transactions on Networking*, 2017.
- [30] L. Wang, Q. Li, Y. Jiang, J. Wu, "Towards Mitigating Link Flooding Attack Via Incremental SDN Deployment", *IEEE ISCC*, 2016.
- [31] M. Caria, A. Jukan, M. Hoffmann, "Divide and Conquer: Partitioning OSPF networks with SDN", *IEEE IM* 2015.
- [32] C.-Y. Chu, K. Xi, M. Luo, H. Chao, "Congestion-aware Single Link Failure Recovery in Hybrid SDN Networks", *IEEE Infocom*, 2015.
- [33] D.K. Hong, Y. Ma, S. Banerjee, Z.M. Mao, "Incremental Deployment of SDN in Hybrid Enterprise and ISP Networks", *ACM SOSR*, 2016.
- [34] H. Xu, J. Fan, J. Wu, C. Qiao, L. Huang, "Joint Deployment and Routing in Hybrid SDNs", *IEEE/ACM IWQoS*, 2017.
- [35] "The evaluation code for DEG, VOL, Modified greedy, and local search used to produce the results for the Abilene topology presented in section VII" [Online]: [goo.gl/EXoZZZ](http://goo.gl/EXoZZZ)
- [36] P. Tune, M. Roughan, "Internet Traffic Matrices: A Primer", *Recent Advances in Networking*, vol. 1, 2013.
- [37] L. Duan, J. Huang, J. C. Walrand, "Economic Analysis of 4G Upgrade Timing", *IEEE Trans. Mob. Comput.*, vol. 14, no. 5, 2015.
- [38] S. Vissicchio, O. Tilmans, L. Vanbever, and J. Rexford, "Central Control over Distributed Routing", in *Prof. of ACM SIGCOMM*, 2015.
- [39] M.S. Bansal, V.C. Venkaiah, "Improved Fully Polynomial time Approximation Scheme for the 0-1 Multiple-choice Knapsack Problem", in *Proc. SIAM Conference on Discrete Mathematics*, 2004.
- [40] A. Cianfrani, M. Listanti, M. Polverini, "Incremental Deployment of Segment Routing Into an ISP Network: a Traffic Engineering Perspective", *IEEE/ACM Transactions on Networking*, 2017.



**Konstantinos Poularakis** obtained the Diploma, and the M.S. and Ph.D. degrees in Electrical Engineering from the University of Thessaly, Greece, in 2011, 2013 and 2016 respectively. In 2014, he was a Research Intern with Technicolor Research, Paris. Currently, he is a Post-Doctoral researcher at Yale University. His research interests lie in the area of network optimization with emphasis on emerging architectures such as software defined networks and content distribution wireless networks. He was the recipient of several awards and scholarships during his studies, from sources including the Greek State Scholarships foundation (2011), the Center for Research and Technology Hellas (2012), the Alexander S. Onassis Public Benefit Foundation (2013) and the Bodossaki Foundation (2016). He also received the Best Paper Award at the IEEE Infocom 2017.



**George Iosifidis** received the Diploma degree in telecommunications engineering from the Greek Air Force Academy in 2000, and the M.S. and Ph.D. degrees in electrical engineering from the University of Thessaly in 2007 and 2012, respectively. He was a Post-Doctoral Researcher with CERTH, Greece, and Yale University, USA. He is currently the Ussher Assistant Professor in Future Networks with Trinity College Dublin, Ireland. His research interests lie in the broad area of wireless networks and network economics.



**Georgios Smaragdakis** received the Diploma degree in electronic and computer engineering from the Technical University of Crete in 2003, and the Ph.D. degree in computer science from Boston University in 2009. In 2008, he was a Research Intern with Telefonica Research. From 2008 to 2014, he was a Senior Researcher with Deutsche Telekom Laboratories. From 2014 to 2017, he was a Marie Curie Fellow at the Massachusetts Institute of Technology (MIT) Computer Science and the Artificial Intelligent Laboratory, and from 2016 to 2018, a Research Affiliate with the MIT Internet Policy Research Initiative. He is currently a Professor with Technical University (TU) Berlin and a Research Collaborator with Akamai Technologies. His research brings a data- and measurement-driven approach to understand the state and improve the resilience of the Internet, as well as to enhance Web privacy. His research received the Best Paper Awards at ACM IMC in 2011, 2016, and 2018, ACM CoNEXT in 2015, and the IEEE INFOCOM in 2017, the Marie Curie International Outgoing Fellowship in 2013, the European Research Council Starting Grant Award in 2015, and an IETF/IRTF Applied Networking Research Prize in 2019.



**Leandros Tassiulas** (S'89, M'91, SM/05 F/07) is the John C. Malone Professor of Electrical Engineering at Yale University. His research interests are in the field of computer and communication networks with emphasis on fundamental mathematical models and algorithms of complex networks, architectures and protocols of wireless systems, sensor networks, novel internet architectures and experimental platforms for network research. His most notable contributions include the max-weight scheduling algorithm and the back-

pressure network control policy, opportunistic scheduling in wireless, the maximum lifetime approach for wireless network energy management, and the consideration of joint access control and antenna transmission management in multiple antenna wireless systems. Dr. Tassiulas is a Fellow of IEEE (2007). His research has been recognized by several awards including the IEEE Koji Kobayashi computer and communications award, the inaugural INFOCOM 2007 Achievement Award "for fundamental contributions to resource allocation in communication networks," the INFOCOM 1994 best paper award, a National Science Foundation (NSF) Research Initiation Award (1992), an NSF CAREER Award (1995), an Office of Naval Research Young Investigator Award (1997) and a Bodossaki Foundation award (1999). He holds a Ph.D. in Electrical Engineering from the University of Maryland, College Park (1991). He has held faculty positions at Polytechnic University, New York, University of Maryland, College Park, and University of Thessaly, Greece.